Improved Analysis of Deterministic Load-Balancing Schemes

Petra Berenbrink, Ralf Klasing, Adrian Kosowski, Frederik Mallmann-Trenn, Przemyslaw Uznanski

Improved Analysis of Deterministic Load-Balancing Schemes

Petra Berenbrink[1], Ralf Klasing[2], Adrian Kosowski[3], Frederik Mallmann-Trenn[1,4], and Przemysław Uznański*[5]

[1]Simon Fraser University, {petra, fmallman}@sfu.ca
[2]CNRS – LaBRI – Université de Bordeaux, ralf.klasing@labri.fr
[3]Inria – LIAFA – Paris Diderot University, adrian.kosowski@inria.fr
[4]École Normale Supérieure, Frederik.Mallmann-Trenn@ens.fr
[5]HIIT – Aalto University, przemyslaw.uznanski@aalto.fi

Abstract

We consider the problem of deterministic load balancing of tokens in the discrete model. A set of n processors is connected into a d-regular undirected network. In every time step, each processor exchanges some of its tokens with each of its neighbors in the network. The goal is to minimize the discrepancy between the number of tokens on the most-loaded and the least-loaded processor as quickly as possible.

Rabani *et al.* (1998) present a general technique for the analysis of a wide class of discrete load balancing algorithms. Their approach is to characterize the deviation between the actual loads of a discrete balancing algorithm with the distribution generated by a related Markov chain. The Markov chain can also be regarded as the underlying model of a continuous diffusion algorithm. Rabani *et al.* showed that after time $T = \mathcal{O}(\log(Kn)/\mu)$, any algorithm of their class achieves a discrepancy of $\mathcal{O}(d \log n/\mu)$, where μ is the spectral gap of the transition matrix of the graph, and K is the initial load discrepancy in the system.

In this work we identify some natural additional conditions on deterministic balancing algorithms, resulting in a class of algorithms reaching a smaller discrepancy. This class contains well-known algorithms, *e.g.*, the ROTOR-ROUTER. Specifically, we introduce the notion of *cumulatively fair* load-balancing algorithms where in any interval of consecutive time steps, the total number of tokens sent out over an edge by a node is the same (up to constants) for all adjacent edges. We prove that algorithms which are cumulatively fair and where every node retains a sufficient part of its load in each step, achieve a discrepancy of $\mathcal{O}(\min\{d\sqrt{\log n/\mu}, d\sqrt{n}\})$ in time $\mathcal{O}(T)$. We also show that in general neither of these assumptions may be omitted without increasing discrepancy. We then show by a combinatorial potential reduction argument that any cumulatively fair scheme satisfying some additional assumptions achieves a discrepancy of $\mathcal{O}(d)$ almost as quickly as the continuous diffusion process. This positive result applies to some of the simplest and most natural discrete load balancing schemes.

*Part of the work was done while the author was affiliated with LIF, CNRS and Aix-Marseille Université

1 Introduction

In this paper, we analyze *diffusion-based load balancing algorithms*. We assume that the processors are connected by an arbitrary d-regular graph G. At the beginning, every node has a certain number of tokens, representing its initial load. In general, diffusion-based load balancing algorithms operate in parallel, in synchronous steps. In each step, every node balances its load with *all* of its neighbors. The goal is to distribute the tokens in the network graph as evenly as possible. More precisely, we aim at minimizing the *discrepancy* (also known as *smoothness*), which is defined as the difference between the maximum load and the minimum load, taken over all nodes of the network.

One distinguishes between *continuous* load balancing models, in which load can be split arbitrarily, and the much more realistic *discrete* model, in which load is modeled by tokens which cannot be split. In the former case, the standard continuous diffusion algorithm works as follows. Every node u having load $x(u)$ considers all of its neighbors at the same time, and sends $x(u)/(d+1)$ load to each of its d neighbors, keeping $x(u)/(d+1)$ load for itself. This process may also be implemented more efficiently: for each neighbor v of u, node u sends exactly $\max\{0, (x(u)-x(v))/(d+1)\}$ load to v. It is well-known (*cf. e.g.* [19]) that the load in the continuous model will eventually be perfectly balanced. In the discrete case with indivisible tokens, exact simulation of the continuous process is not possible. A node u may instead try to round the amount of load sent to its neighbor up or down to an integer. Discrete balancing approaches are, in general, much harder to analyze than continuous algorithms.

In [17], Rabani *et al.* suggest a framework to analyze a wide class of discrete neighborhood load balancing algorithms in regular graphs (it can be adapted to non-regular graphs). The scheme compares the discrete balancing algorithm with its continuous version, and the difference is used to bound the so-called error that occurs due to the rounding. Their results hold for *round-fair* algorithms where any node u having load $x_t(u)$ at time t sends either $\lfloor x_t(u)/(d+1) \rfloor$ or $\lceil x_t(u)/(d+1) \rceil$ tokens over its edges. They show that the discrepancy is bounded by $\mathcal{O}(d \log n/\mu)$ after $T = \mathcal{O}(\log(Kn)/\mu)$ steps, where μ is the eigenvalue gap of the transition matrix of the underlying Markov chain and K is the initial load discrepancy. The scheme applies to any discrete load balancing scheme which, at every time step, rounds the load which would be exchanged in the continuous diffusion process by a given pair of nodes to one of the nearest integers, either up or down.

The time T in the above bound is also the time in which a continuous algorithm balances the system load (more or less) completely. T is closely related to the *mixing time* of a random walk on G, which is the time it takes for a random walk to be on every node with almost the same probability. Within the class of schemes considered in [17], the bound of $\mathcal{O}(d \log n/\mu)$ on discrepancy cannot be improved for many important graph classes, such as constant-degree expanders. Since the work [17], many different refinements and variants of this approach have been proposed [4, 9, 10, 18], as well as extensions to other models, including systems with non-uniform tokens [4] and non-uniform machines [2].

1.1 Our Contribution

The main goal of this paper is to continue the work of [17] by analyzing properties/classes of deterministic algorithms that balance better in the diffusive model than the class defined in [17]. We suggest and analyze two general classes of balancing algorithms (called cumulatively fair balancers and good balancers) that include many well-known diffusion algorithms and we bound the discrepancy they achieve after $\mathcal{O}(T)$ time steps. Within the framework of schemes which are deterministic and pose no major implementation challenges (i.e., do not generate negative load and do not rely on additional communication), we obtain a number of significant improvements with respect to the state-of-the-art, cf. Table 1.

For our algorithms we assume that every node of the graph has in addition to its d original edges $d^\circ \geq d$ many self-loops. We define $d^+ = d^\circ + d$ as the degree of the graph including the self-loops.

Algorithm	Discrepancy after time $\mathcal{O}(T)$	Time to reach $O(d)$ discrepancy	Ref.	D	SL	NL	NC
Discrete diffusion with arbitrary rounding on edges	$\mathcal{O}\left(\frac{d\log n}{\mu}\right)$	✗ (in general)	[17]	(✓)	(✓)	(✓)	(✓)
Randomized distribution of extra tokens by vertices	$\mathcal{O}\left(\min\left\{d^2, d+\sqrt{\frac{d\log d}{\mu}}\right\}\sqrt{\log n}\right)$	✗	[5,18]	✗	✓	✓	✓
Randomized rounding to nearest integers on edges	$\mathcal{O}\left(\sqrt{d\log n}\right)$	✗	[18]	✗	✓	✗	✓
Computation based on continuous diffusion	$\Theta(d)$	$\mathcal{O}(T)$	[4]	✓	✗	✗	✗
Cumulatively fair balancers	$\mathcal{O}\left(d\min\left\{\sqrt{\frac{\log n}{\mu}}, \sqrt{n}\right\}\right)$	✗ (in general)	Thm 2.3	(✓)	(✓)	(✓)	(✓)
• ROTOR-ROUTER	"	✗	Thm 2.3	✓	✗	✓	✓
• SEND ($\lfloor x/d^+ \rfloor$)	"	open	Thm 2.3	✓	✓	✓	✓
• **Good s-balancers**	"	$\mathcal{O}\left(T+\frac{d\log^2 n}{\mu}\right)$	Thm 3.3	(✓)	(✓)	(✓)	(✓)
• ROTOR-ROUTER*	"	"	Thm 3.3	✓	✗	✓	✓
• SEND ($[x/d^+]$), for any $d^+ > 2d$	"	"	Thm 3.3	✓	✓	✓	✓
• SEND ($[x/d^+]$), for any $d^+ \geq 3d$	"	$\mathcal{O}\left(T+\frac{\log^2 n}{\mu}\right)$	Thm 3.3	✓	✓	✓	✓

Legend: D — Deterministic process; SL — Stateless process; NL — Cannot produce negative loads; NC — No additional communication required; (✓) denotes properties achieved for some implementations; $[\cdot]$ denotes rounding to the nearest integer.

Table 1: A comparison of the discrepancy of load-balancing algorithms in the diffusive model for d-regular graphs. In all result statements, if not specified otherwise, we assume that the graph is augmented with at least d self-loops per node.

Cumulatively δ-fair balancers. We call an algorithm cumulatively δ-fair if *(i)* for every interval of consecutive time steps, the total number of tokens an algorithm sends over the original edges (non-self-loops) differs by at most a small constant δ and *(ii)* for every time step every edge (original edges and self-loops) of a node receives at least $\lfloor x/d^+ \rfloor$ many tokens, where x is the current load of the node. Cumulatively δ-fair balancers are a subclass of the algorithms studied in [17], which satisfied weaker fairness conditions. We show (Theorem 2.3) that algorithms satisfying these conditions with $d^\circ \geq d$ achieve a discrepancy of $\mathcal{O}(d \cdot \min\{\sqrt{\log n/\mu}, \sqrt{n}\})$ in $\mathcal{O}(T)$ time steps. The restrictions result in deterministic balancing schemes with an improved discrepancy after $\mathcal{O}(T)$ time steps. For example, for expanders, the achieved discrepancy after time $\mathcal{O}(T)$ is $\mathcal{O}(\sqrt{\log n})$, as opposed to $\Theta(\log n)$.

Additionally to the upper bound, we show that the discrepancy can be of order $\Omega(d \cdot \mathrm{diam})$ (diam is the diameter of the graph) if we drop the condition of cumulative fairness (Theorem 4.1) or remove self-loops completely (Theorem 4.3). In more detail, Theorem 4.1 shows that there are round-fair balancers that satisfy the constraints of [17] and that have $\Omega(d \cdot \mathrm{diam})$ discrepancy nonetheless. Such discrepancy is worse than the one we obtain for cumulatively fair balancers (Theorem 2.3) for many graph classes. In Theorem 4.3 we show for a specific cumulatively 1-fair balancer not using any self-loops (ROTOR-ROUTER), that it cannot achieve discrepancy better than $\Omega(n)$ on a cycle with n nodes.

The class of cumulatively fair balancers contains many well-known deterministic algorithms. In particular, many of these algorithms are *stateless*, meaning the load any node sends over edges in any step depends solely on the load of the node at this time step. An example for a stateless cumulatively fair balancer is SEND ($\lfloor x/d^+ \rfloor$), where a node with load x sends $\lfloor x/d^+ \rfloor$ many tokens to every original edge (non-self-loop). The remaining $x - \lfloor x/d^+ \rfloor$ tokens are distributed over self-loops such that every self-loop receives at least $\lfloor x/d^+ \rfloor$ many tokens. Similarly, the algorithm SEND ($[x/d^+]$) is cumulatively fair. Here $[x/d^+]$ rounds x/d^+ to the next integer and a node with load x sends $[x/d^+]$ tokens over every original edge. Another well-known algorithm in this class is the so-called ROTOR-ROUTER model (also referred to in the literature as the Propp model, [3,6,11]) which uses a simple round-robin approach to distribute the tokens to the d^+ neighbors, i.e., over all its edges.

Good s-balancers. The class of good s-balancers can be regarded as a restriction of cumulatively 1-fair balancers. A cumulatively 1-fair balancer is a good s-balancer if it *(i)* is round-fair (every edge receives

$\lfloor x/d^+ \rfloor$ or $\lceil x/d^+ \rceil$ many tokens, where x is the current load of a node) *(ii)* sends over at least s self-loops $\lceil x/d^+ \rceil$ many tokens in every round. We show that algorithms of this class achieve a $\mathcal{O}(d)$-discrepancy within $\mathcal{O}(T + d \log^2 n/\mu)$ time steps. Moreover, if $s = \Omega(d)$, then the same discrepancy of $\mathcal{O}(d)$ is reached within $\mathcal{O}(T + \log^2 n/\mu)$ time steps. The class of good s-balancers contains many well-known algorithms. For example, the algorithm SEND $([x/d^+])$ is a good s-balancer for $d^+ > 2d$. If $d^+ \geq 3d$, then it achieves a $\mathcal{O}(d)$ discrepancy in $\mathcal{O}(T + \log^2 n/\mu)$ time. Additionally, the class contains some variants of the rotor-router approach, *e.g.*, one variant which we denote by ROTOR-ROUTER$*$. This algorithm maintains $d - 1$ self-loops, together with one special self-loop, which always receives $\lceil x_t(u)/(2d) \rceil$ tokens. The remaining tokens are distributed fairly using a rotor-router on the original edges and the $d - 1$ self-loops.

The only other result in the literature known to the authors which achieves a discrepancy of $\mathcal{O}(d)$ in $\mathcal{O}(T)$ steps in the *diffusive model* is the one of [4] (see [18] for $\mathcal{O}(d)$ discrepancy in the dimension exchange model where nodes balance with only one neighbor per round). Their algorithms simulate and mimic (with the discrete tokens) the continuous flow.

Technical contributions. Our techniques for the analysis of cumulatively fair balancers rely on a comparison between our discrete process and the continuous process. The latter can also be regarded as a Markovian process (random walk) which is governed by the transition matrix of the graph. We calculate the total deviation (of any cumulatively fair balancer) to the continuous process as done in [17]. However, instead of doing it step-by-step as in [17], the comparison is done over long time intervals. This was done in, *e.g.*, [13, 21], in the context of the graph exploration problem. Our analysis of this class connects this deviation to the value $\mathcal{O}(\log n/\mu)$, which is a natural upper bound on the mixing time. For the analysis of good s-balancers we combine the algebraic techniques used for the analysis of cumulatively fair balancers with a potential function approach. Whereas all cumulatively 1-fair balancers admit a natural potential function, which is (weakly) monotonous throughout the balancing process, for the narrower class of good s-balancers we can define time phases so that in each time phase, the process exhibits a strict potential drop in each phase of balancing, up to a balancing discrepancy of $\mathcal{O}(d)$. Even though we limit ourselves to regular graphs in this paper, our results can be extended to non-regular graphs.

1.2 Related Work

Herein we only consider related results for load balancing in the discrete diffusive and balancing circuit models, and some results for the rotor-router model which are relevant to our work.

Diffusive load balancing. Discrete load balancing has been studied in numerous works since [17]. The authors of [9] propose a deterministic load balancing process in which the continuous load transferred along each edge is rounded up or down deterministically, such that the sum of the rounding errors on each edge up to an arbitrary step t is bounded by a constant. This property is called the *bounded-error property*. Then they show that after T steps any process with bounded-error property achieves a discrepancy of $\mathcal{O}(\log^{3/2} n)$ for hypercubes and $\mathcal{O}(1)$ for constant-degree tori. There are no similar results for other graph classes. Note that the algorithm of [9] has the problem that the original demand of a node might exceed its available load, leading to so-called *negative load*.

In [3], the authors consider ROTOR-ROUTER-type walks as a model for load balancing. It is assumed that half of the edges of every node are self-loops. The authors present an algorithm which falls in the class of *bounded-error diffusion processes* introduced in [9]. This results in discrepancy bounds of $\mathcal{O}(\log^{3/2} n)$ and $\mathcal{O}(1)$ for hypercube and r-dimensional torus with $r = \mathcal{O}(1)$. In [2], the authors consider the diffusion algorithms that always round down for heterogeneous networks. They also show that a better load balance can be obtained when the algorithm is allowed to run longer than T steps.

In [4], the authors propose an algorithm that achieves discrepancy of $2d$ after T steps for any graph. For every edge e and step t, their algorithm calculates the number of tokens that should be sent over e in t such that the total number of tokens forwarded over e (over the first t steps) stays as close as possible

to the amount of load that is sent by the continuous algorithm over e during the first t steps. However, their algorithm can result in negative load when the initial load of any node is not sufficiently large and it has to calculate the number of tokens that the continuous algorithm sends over all edges. Note that the algorithm presented in this paper has to simulate the continuous algorithm in order to calculate the load that it has to transfer over any edge, whereas our algorithms are much easier and they do not need any additional information, not even the load of their neighbors.

There are several publications that suggest randomized rounding schemes [1, 3–5, 9, 18] to convert the continuous load that is transferred over an edge into discrete load. The algorithm of [5] calculates the number of *additional* tokens (the difference between the continuous flow forwarded over edges and the number of tokens forwarded by the discrete algorithm after rounding *down*). All additional tokens are sent to randomly chosen neighbors. Their discrepancy bounds after T steps are $\mathcal{O}(d \log\log n/\mu)$ and $\mathcal{O}(d\sqrt{\log n} + \sqrt{d \log n \log d/\mu})$ for d-regular graphs, $\mathcal{O}(d \log\log n)$ for expanders, $\mathcal{O}(\log n)$ for hypercubes, and $\mathcal{O}(\sqrt{\log n})$ for tori. The authors of [18] present two randomized algorithms for the diffusive model. They achieve after $\mathcal{O}(T)$ time a discrepancy of $\mathcal{O}(d^2\sqrt{\log n})$ by first sending $\lfloor x(u)/(d+1) \rfloor$ tokens to every neighbor and itself, and afterwards by distributing the remaining tokens randomly. Additionally, they provide an algorithm which achieves after $\mathcal{O}(T)$ time a discrepancy of $\mathcal{O}(\sqrt{d \log n})$ by rounding the flow sent over edges randomly to the nearest integers which might cause negative loads. For a comparison with our results see Table 1.

Dimension exchange model. In the Dimension Exchange model, the nodes are only allowed to balance with one neighbor at a time. Whereas for all diffusion algorithms considered so far the discrepancy in the diffusion model is at least d, dimension exchange algorithms are able to balance the load up to an additive constant. In [10] the authors consider a discrete dimension exchange algorithm for the matching model. Every node i that is connected to a matching edge calculates the load difference over that edge. If that value is positive, the algorithm rounds it up or down, each with probability one half. This result is improved in [18], where the authors show that a constant final discrepancy can be achieved within $\mathcal{O}(T)$ steps for regular graphs in the random matching model, and constant-degree regular graphs in the periodic matching (balancing circuit) model.

Rotor-router walks. Originally introduced in [16], the *rotor-router walk* model was employed by Jim Propp for derandomizing the random walk, thereby frequently appearing under the alternative names of *Propp machines* and *deterministic random walks* [6, 8, 11, 12]. In the rotor-router model, the nodes send their tokens out in a round-robin fashion. It is assumed that the edges of the nodes are cyclically ordered, and that every node is equipped with a rotor which points to one of its edges. Every node first sends one token over the edge pointed to by the rotor. The rotor is moved to the next edge which will be used by the next token, and so on, until all tokens of the node have been sent out over one of the edges. It has been shown that the rotor walks capture the average behaviour of random walks in a variety of respects such as hitting probabilities and hitting times. The rotor-router model can be used for load balancing, and directly fits into the framework we consider in this paper. The authors of [20] obtain rough bounds on the discrepancy, independently of [17]. In [3], the authors study a lazy version of the rotor-router process (half of the edges are self-loops) for load balancing. They prove that the rotor walk falls in the class of *bounded-error diffusion processes* introduced in [9]. Using this fact they obtain discrepancy bounds of $\mathcal{O}(\log^{3/2} n)$ and $\mathcal{O}(1)$ for the hypercube and r-dimensional torus with $r = \mathcal{O}(1)$, respectively, which improve the best existing bounds of $\mathcal{O}(\log^2 n)$ and $\mathcal{O}(n^{1/r})$ in this graph class.

1.3 Model and Notation

In this section we define our general model, which applies to both classes of studied algorithms.

The input of the load-balancing process is a symmetric and directed regular graph $G = (V, E)$ with n nodes. Every node has out-degree and in-degree d. We have $m \in \mathbf{N}$ indivisible tokens (workload) which are

arbitrarily distributed over the nodes of the network. For simplicity of notation only, we assume that initially G does not contain multiple edges. In general, the nodes of the graph may be treated as anonymous, and no node identifiers will be required. The time is divided into synchronized steps. Let $\mathbf{x}_t = (x_t(1), \ldots, x_t(n))$ be the *load vector* at the beginning of step t, where $x_t(u)$ corresponds to the load of node u at the beginning of step t. In particular, $\mathbf{x}_1$ denotes the initial load distribution and $\bar{\mathbf{x}}$ is the real-valued vector resulting when every node has achieved average load, $\bar{\mathbf{x}}(u) = \frac{1}{n}\sum_{u\in V} x_1(u) \equiv \bar{x}$, for all $u \in V$. Note that the total load summed over all nodes does not change over time. The *discrepancy* is defined as the load difference between the node with the *highest load* and the node with the *lowest load*. We will denote by K the maximal initial discrepancy in $\mathbf{x}_1$, *i.e.*, $K = \max_{u\in V} x_1(u) - \min_{u\in V} x_1(u)$. The *balancedness* of an algorithm is defined as the gap between the node with the highest load and the *average load*.

In order to introduce self-loops, we transform G into the graph $G^+ = (V, E \cup E^\circ)$ by adding d° self-loops to every node. For a fixed node $u \in V$, let $E_u^\circ = \{e_1(u,u), \ldots, e_{d^\circ}(u,u)\}$ denote the set of self-loops of u and let E_u denote the original edges of u in G. We assume $d^\circ = \mathcal{O}(d)$. We can now define $E_u^+ = E_u^\circ \cup E_u$ and $E^\circ = \bigcup_{u\in V} E_u^\circ$. In the following, we call G the *original graph* and G^+ the *balancing graph*. The edges E° are called self-loop edges of G^+ and E_u are the original edges. We remark again that the balancing graph is introduced for purposes of analysis, only, and is completely transparent from the perspective of algorithm design. We also define $d^+ = d + d^\circ$ as the degree of any node in G^+. $N(u)$ is the set of direct neighbors of u in G^+, *i.e.*, it contains all neighbors of u in G including u itself (because of the self-loops).

For a fixed edge $e = (u, v) \in E^+$ let $f_t(e)$ be the number of tokens which u sends to v in step t. In particular, let $f_t(u,u) = \sum_{e\in E_u^\circ} f_t(e)$. Let $F_t(e)$ denote the cumulative load sent from u to v in steps $1, \ldots, t$, i.e, $F_t(e) = \sum_{\tau\le t} f_\tau(e)$. For a fixed node u, we define $f_t^{out}(u) = \sum_{v\in N(u)} f_t(u,v)$ to be the number of tokens (or flow) leaving u in step t. The incoming flow is then defined as $f_t^{in}(u) = \sum_{\{v:u\in N(v)\}} f_t(v,u)$. We define the cumulative incoming and original flows $F_t^{out}(u)$ and $F_t^{in}(u)$ accordingly, and $\mathbf{F}_t^{out}$ and $\mathbf{F}_t^{in}$ are defined as the vectors of the (cumulative) original and incoming flow. We will say that edge $e = (u, v)$ received x tokens when node u sends x tokens to v.

2 Results for Cumulatively Fair Balancers

In this section we present a general class of algorithms, called *cumulative fair* algorithms, and analyze their discrepancy after $T = \mathcal{O}\left((\log K + \log n)/\mu\right)$ many time steps. Note that T is the balancing time of the continuous diffusion algorithm if the initial discrepancy is K, and d is the number of original edges (non-self-loops) of each node.

We call an algorithm *cumulatively fair* if the flow that is sent out over every edge of u (including the self-loops) up to step t can differ by at most δ.

Definition 2.1. *Let δ be a constant. An algorithm is called* cumulatively δ-fair *if for all $t \in \mathbf{N}$, $u \in V$*

- *every edge $e \in E_u^+$ receives at least $\lfloor x_t(u)/d^+ \rfloor$ many tokens.*
- *all original edges $e_1, e_2 \in E_u$ satisfy $|F_t(e_1) - F_t(e_2)| \le \delta$.*

Note that *round-fair* algorithms (defined in [17]) are not necessarily cumulatively δ-fair for any fixed δ.

Observation 2.2. *The algorithms* SEND $(\lfloor x/d^+ \rfloor)$ *and* SEND $(\lceil x/d^+ \rceil)$ *are* 0*-cumulatively fair. Furthermore,* ROTOR-ROUTER *is cumulatively* 1*-fair.*

For cumulatively fair balancers we show the following bound.

Theorem 2.3. *Let G be any d-regular input graph and let d^+ is the degree of the balancing graph G^+ (including the self-loops). Let δ be a constant. Assume A is a cumulatively δ-fair balancer. Then, after $\mathcal{O}\left(\log(Kn)/\mu\right)$ time steps, A achieves a discrepancy of*

(i) $\mathcal{O}\left((\delta+1)\cdot d\cdot\sqrt{\log n/\mu}\right)$ *for* $d^+ \geq 2d$.
(ii) $\mathcal{O}\left((\delta+1)\cdot d\cdot\sqrt{n}\right)$ *for* $d^+ \geq 2d$.
(iii) $\mathcal{O}\left((\delta+1)\cdot d\cdot\log n/\mu\right)$ *for arbitrary* $d^+ \geq d+1$.

For constant δ and at least d self-loops, the results of Claim (i) of the above theorem show a better discrepancy after T steps compared to the result of [17]. Claim (ii) provides an improvement for graphs with a bad expansion (small eigenvalue gap), such as cycles.

The rest of this section is devoted to the proof of Theorem 2.3. The core idea of the proof is to regard the balancing process over several steps and to observe that the cumulative load of the nodes is closely related to a random walk of all tokens. The difference to the random walk can be bounded by a small corrective vector depending on δ. We start by providing some additional definitions.

In the analysis, we change the way the tokens are kept at a node: In addition to sending tokens over self-loop edges we allow a node to retain a remainder (of tokens) of size $r < d^+$ at every node. The reason for this is that the proof requires the cumulative fairness on all edges and not just on original edges. One can show (Proposition A.2) that every cumulatively fair balancer can be transformed (by shifting tokens from self-loops to the remainder) into an algorithm guaranteeing cumulative fairness on all edges and that this transformed algorithm sends exactly the same load over original edges in every round.

Let the remainder $r_t(u)$ of node u in step t be the number of tokens of u that will not participate in the load distribution over its original edges and self-loops. Then, $\mathbf{r}_t = (r_t(1),\ldots,r_t(n))$ denotes the *remainder vector* at step t, where $r_t(i)$ is the number of tokens kept by the i'th node of G in step t. We will denote by r the upper bound on the maximum remainder of an algorithm, satisfying $|r_t(u)| \leq r \leq d^+$ for every time step t and every $u \in V$. For the ease of notation, we assume that $r < d^+$. Note that for all $u \in V$ and all steps t

$$x_1(u) + F^{in}_{t-1}(u) = r_t(u) + F^{out}_t(u). \tag{1}$$

Moreover,

$$x_t(u) = f^{out}_t(u) + r_t(u). \tag{2}$$

Let $\mathbf{P}$ denote the transition matrix of a random walk of G^+. Let $\mathbf{P}(u,v)$ denote the one-step probability for the walk to go from u to v. Then $\mathbf{P}(u,v) = 1/d^+$ if $(u,v) \in E$, $\mathbf{P}(u,v) = d^\circ/d^+$ if $u = v$, and $\mathbf{P}(u,v) = 0$ otherwise.

Let μ be the eigenvalue gap of $\mathbf{P}$, i.e., $\mu = 1-\lambda_2$, where λ_2 is the second largest eigenvalue. We define $\mathbf{P}^t$ to be the t-steps transition matrix, *i.e.*, $\mathbf{P}^t = \mathbf{P}\cdot\mathbf{P}^{t-1}$. We define the *steady-state* distribution as $\mathbf{P}^\infty = \lim_{t\to\infty}\mathbf{P}^t$. Note that $\forall u,v \in V$, $\mathbf{P}^\infty(u,v) = d^+/(2|E^+|) = 1/n$. Observe that $P^\infty\cdot\mathbf{x}_1 = (\bar{x},\bar{x},\ldots,\bar{x})$. We can express $\mathbf{P}^t$ as $\mathbf{P}^t = \mathbf{P}^\infty + \mathbf{\Lambda}_t$, where $\mathbf{\Lambda}_t$ is the error-term calculating the difference between $\mathbf{P}^t$ and the steady-state distribution. Let $p \geq 1$. The p-norm of a vector $\mathbf{r}$ is defined as $\|\mathbf{r}\|_p = (\sum_{i=1}^n |r_i|^p)^{\frac{1}{p}}$. In particular, $\|\mathbf{r}\|_\infty$ is defined to be $\max\{|r_1|,\ldots,|r_n|\}$.

We are ready to prove the main theorem of this section. The core idea of the proof is to calculate the total deviation between any cumulatively fair balancer and a continuous process, similar to [17]. However, instead of comparing the two processes step-by-step as in [17], the comparison is done over long time intervals similar to [13]. This deviation is then connected to the value $\mathcal{O}(\log n/\mu)$.

Proof of Theorem 2.3. Fix a node $u \in V$. Note that by the definition of cumulatively δ-fairness and Proposition A.2 we have for all $(u,v) \in E^+_u$ that

$$\left|F_t(u,v) - F^{out}_t(u)/d^+\right| \leq \delta. \tag{3}$$

We will define a corrective vector which at time t measures the difference between the load the nodes sent over original edges at time t and the load the nodes should have sent over these edges in order to ensure

that every original edge received the exact same (continuous) load until time t. Formally, we define the n-dimensional corrective vector $\boldsymbol{\delta}_{t,u}$ with $\delta_{t,u}(v) = F_t(v,u) - F_t^{out}(v)/d^+$ for $v \neq u$ and $\delta_{t,u}(u) = F_t(u,u) - d^\circ/d^+ \cdot F_t^{out}(u)$. The entries of the corrective vector satisfy $|\delta_{t,u}(v)| \leq \delta$ for $v \in N(u) \setminus \{u\}$, $|\delta_{t,u}(u)| \leq d^\circ\delta$, and $\delta_{t,u}(v) = 0$ for $v \notin N(u)$. Consequently, $\|\boldsymbol{\delta}_{t,u}\|_1 \leq \delta d^+$. Then we derive from (3) the following bound on the incoming cumulative load of node u

$$\begin{aligned} F_t^{in}(u) &\underset{def.}{=} \sum_{v \in N(u)} F_t(v,u) = \sum_{v \in N(u)\setminus\{u\}} F_t(v,u) + F_t(u,u) \\ &\underset{(3)}{=} \sum_{v \in N(u)\setminus\{u\}} \left(\frac{1}{d^+} F_t^{out}(v) + \delta_{t,u}(v) \right) + \frac{d^\circ}{d^+} F_t^{out}(u) + \delta_{t,u}(u) \\ &= \sum_{v \in N(u)\setminus\{u\}} \frac{1}{d^+} F_t^{out}(v) + \frac{d^\circ}{d^+} F_t^{out}(u) + \|\boldsymbol{\delta}_{t,u}\|_1 . \end{aligned} \tag{4}$$

Rewriting (1) by introducing (4) we get:

$$F_t^{out}(u) = x_1(u) + \sum_{v \in N(u)\setminus\{u\}} \frac{1}{d^+} F_{t-1}^{out}(v) + \frac{d^\circ}{d^+} F_{t-1}^{out}(u) + \underbrace{\|\boldsymbol{\delta}_{t,u}\|_1 - r_t(u)}_{\varepsilon_t(u)} . \tag{5}$$

We have $\|\varepsilon_t(u)\|_\infty \leq \delta d^+ + r$. Rewriting (5) in vector form, we obtain

$$\mathbf{F}_t^{out} = \mathbf{x}_1 + \mathbf{P} \cdot \mathbf{F}_{t-1}^{out} + \boldsymbol{\varepsilon}_t = \sum_{0 \leq \tau < t} \mathbf{P}^\tau \mathbf{x}_1 + \sum_{1 \leq \tau \leq t} \mathbf{P}^{t-\tau} \cdot \boldsymbol{\varepsilon}_\tau .$$

For any $\widehat{T} > 0$, the number of tokens leaving node u in the interval $[t+1; t+\widehat{T}]$ is $\mathbf{F}_{t+\widehat{T}}^{out}(u) - \mathbf{F}_t^{out}(u)$. Hence,

$$\mathbf{F}_{t+\widehat{T}}^{out} - \mathbf{F}_t^{out} = \sum_{t \leq \tau < t+\widehat{T}} \mathbf{P}^\tau \mathbf{x}_1 + \sum_{1 \leq \tau \leq t} (\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau}) \cdot \boldsymbol{\varepsilon}_\tau + \sum_{t < \tau \leq t+\widehat{T}} \mathbf{P}^{t+\widehat{T}-\tau} \cdot \boldsymbol{\varepsilon}_\tau .$$

We set $t^* = t - 4t_\mu$, where $t_\mu = 6 \log n/\mu$, and substitute $\mathbf{P}^\tau = \mathbf{P}^\infty + \boldsymbol{\Lambda}_\tau$. We derive

$$\begin{aligned} \mathbf{F}_{t+\widehat{T}}^{out} - \mathbf{F}_t^{out} &= \sum_{t \leq \tau < t+\widehat{T}} \mathbf{P}^\tau \mathbf{x}_1 + \sum_{1 \leq \tau \leq t^*} (\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau}) \boldsymbol{\varepsilon}_\tau + \sum_{t^* < \tau \leq t} (\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau}) \boldsymbol{\varepsilon}_\tau + \sum_{t < \tau \leq t+\widehat{T}} \mathbf{P}^{t+\widehat{T}-\tau} \boldsymbol{\varepsilon}_\tau \\ &= \widehat{T}\mathbf{P}^\infty \mathbf{x}_1 + \sum_{t \leq \tau < t+\widehat{T}} \boldsymbol{\Lambda}_\tau \mathbf{x}_1 \quad + \sum_{1 \leq \tau \leq t^*} (\boldsymbol{\Lambda}_{t+\widehat{T}-\tau} - \boldsymbol{\Lambda}_{t-\tau}) \boldsymbol{\varepsilon}_\tau + \sum_{t^* < \tau \leq t} (\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau}) \cdot \boldsymbol{\varepsilon}_\tau + \sum_{t < \tau \leq t+\widehat{T}} \mathbf{P}^{t+\widehat{T}-\tau} \cdot \boldsymbol{\varepsilon}_\tau . \end{aligned} \tag{6}$$

In the following we use $\bar{\mathbf{x}} = \mathbf{P}^\infty \mathbf{x}_1$.

$$\begin{aligned} \left\| \sum_{t < \tau \leq t+\widehat{T}} \mathbf{x}_\tau - \widehat{T} \cdot \bar{\mathbf{x}} \right\|_\infty &\underset{(2)}{=} \left\| \left(\mathbf{F}_{t+\widehat{T}}^{out} - \mathbf{F}_t^{out} + \sum_{t < \tau \leq t+\widehat{T}} \mathbf{r}_\tau \right) - \widehat{T}\mathbf{P}^\infty \mathbf{x}_1 \right\|_\infty \\ &\underset{(6)}{\leq} \sum_{t \leq \tau < t+\widehat{T}} \|\boldsymbol{\Lambda}_\tau \mathbf{x}_1\|_\infty \quad + \sum_{1 \leq \tau \leq t^*} \left(\left\| \boldsymbol{\Lambda}_{t+\widehat{T}-\tau} \boldsymbol{\varepsilon}_\tau \right\|_\infty + \|\boldsymbol{\Lambda}_{t-\tau} \boldsymbol{\varepsilon}_\tau\|_\infty \right) \\ &\quad + \sum_{t^* < \tau \leq t} \left\| (\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau}) \boldsymbol{\varepsilon}_\tau \right\|_\infty \quad + \sum_{t < \tau \leq t+\widehat{T}} \left\| \mathbf{P}^{t+\widehat{T}-\tau} \boldsymbol{\varepsilon}_\tau \right\|_\infty \quad + \sum_{t < \tau \leq t+\widehat{T}} \|\mathbf{r}_\tau\|_\infty . \end{aligned}$$

By well-known properties of mixing in graphs (cf. claims (i) and (ii) of Lemma A.1 in the Appendix) it follows for $t \geq 16 \cdot \log(nK)/\mu$ that $\forall_{\tau \geq t} \left\| \mathbf{\Lambda}_\tau \mathbf{x}_1 \right\|_\infty \leq 2^{-4}$, and moreover $\sum_{\tau \geq 4 \cdot t_\mu} \left\| \mathbf{\Lambda}_\tau \boldsymbol{\varepsilon}_\tau \right\|_\infty \leq n^{-4} \cdot \max_{\tau \geq 4 \cdot t_\mu} \{ \left\| \boldsymbol{\varepsilon}_\tau \right\|_\infty \}$. This results in

$$\begin{aligned}\left\| \sum_{t < \tau \leq t+\widehat{T}} \mathbf{x}_\tau - \widehat{T} \cdot \bar{\mathbf{x}} \right\|_\infty &\leq \widehat{T} \cdot 2^{-4} \ + 2 \cdot n^{-4} \max_{1 \leq \tau \leq t^*} \left\| \boldsymbol{\varepsilon}_\tau \right\|_\infty + \sum_{t^* < \tau \leq t} \left\| (\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau}) \boldsymbol{\varepsilon}_\tau \right\|_\infty \\ &\quad + \widehat{T}(\delta d^+ + r) \ + \widehat{T} r \\ &\leq \frac{\widehat{T}}{4} + (\delta d^+ + r) + \sum_{t^* < \tau \leq t} \left\| (\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau}) \boldsymbol{\varepsilon}_\tau \right\|_\infty + \widehat{T}(\delta d^+ + 2r).\end{aligned}$$

Dividing the last equation by $\widehat{T}$ yields

$$\left\| \frac{\sum_{t < \tau \leq t+\widehat{T}} \mathbf{x}_\tau}{\widehat{T}} - \bar{\mathbf{x}} \right\|_\infty \leq \frac{1}{4} + (\delta d^+ + 2r) + \frac{(\delta d^+ + r) + \sum_{t^* < \tau \leq t} \left\| (\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau}) \boldsymbol{\varepsilon}_\tau \right\|_\infty}{\widehat{T}}. \tag{7}$$

In this way, in (7) we have derived a bound on the difference between the average load of a node during an interval of length $\widehat{T}$ and the average load $\bar{x}$.

In the remainder we bound (7) for $\widehat{T} = 1$, which describes the load difference to the average. Fix an arbitrary $t \geq 16 \log(nK)/\mu$. We define $P_t(u, w)$ to be the probability that a random walk following matrix $\mathbf{P}$, initially located at $u \in V$, is located at w after t time steps. We then obtain the following bound (see Appendix A.1 for details):

$$\left\| (\mathbf{P}^{t+1-\tau} - \mathbf{P}^{t-\tau}) \boldsymbol{\varepsilon}_\tau \right\|_\infty \leq (\delta d^+ + r) \cdot \max_{w \in V} \sum_{v \in V} \left| P_{t+1-\tau}(v, w) - P_{t-\tau}(v, w) \right|. \tag{8}$$

Combining (7) and (8) for $\widehat{T} = 1$, recalling that $t^* = t - 4t_\mu = t - 24 \log n/\mu$, and introducing the notation $a = t - \tau$, we obtain:

$$\left\| \mathbf{x}_{t+1} - \bar{\mathbf{x}} \right\|_\infty \leq \frac{1}{4} + r + (\delta d^+ + r) \left(2 + \sum_{a=0}^{24 \log n/\mu} \max_{w \in V} \sum_{v \in V} \left| P_{a+1}(v, w) - P_a(v, w) \right| \right). \tag{9}$$

Thus, the right-hand side of the above expression provides an asymptotic upper bound on the discrepancy at time $O(\log(nK)/\mu)$. It remains to provide an estimate of the sums which appear in the expression. These sums can be analyzed using techniques for bounding probability change (current) of a reversible random walk in successive time steps. In Appendix A.1, we provide three different ways of bounding the expression, leading directly to claims (i), (ii), and (iii) of the theorem.

We remark that it is not clear whether the obtained bounds on the right hand side of (9) are asymptotically tight. For example, the question whether it may be possible to replace "$\sqrt{n}$" in claim (ii) by a term which is polylogarithmic in n is an interesting open question in the theory of random walks on graphs (cf. [15] for some recent related results in the area). □

3 Results for Good s-Balancers

In this section, we consider a subclass of cumulatively 1-fair balancers that achieve a better discrepancy compared to the cumulatively fair balancers if the runtime is slightly larger than T. Algorithms of this class are, by definition, a subclass of cumulatively 1-fair balancers. Hence, Theorem 2.3 also applies to Good s-Balancing Algorithms. A cumulatively 1-fair balancer is also a good s-balancer if the algorithm is *(i)* round-fair and *(ii) self-preferring*, *i.e.*, if it favors self-loop edges over original edges. As we will see in Theorem 3.3, the "more self-preferring" an algorithm is, the faster it balances.

Definition 3.1. *Assume δ is an arbitrary constant and $1 \leq s \leq d^\circ$. An algorithm is called a* good s-balancer *if if for $t \in \mathbf{N}$ and $u \in V$ all edges of u (including self-loops) receive $\lfloor x_t(u)/d^+ \rfloor$ many tokens in step t. The remaining $e(u) = x_t(u) - d^+ \cdot \lfloor x_t(u)/d^+ \rfloor$ have to be distributed over the edges such that*

1. *the algorithm is cumulatively 1-fair*
2. *at least $\min\{s, e(u)\}$ self-loops receive $\lceil x_t(u)/d^+ \rceil$ many tokens. (s-self-preferring)*
3. *every edge $e \in E_u^+$ receives at most $\lceil x_t(u)/d^+ \rceil$ many tokens.*

Note that, by definition, every good s-balancer is round-fair, meaning that every edge receives either $\lfloor x_t(u)/d^+ \rfloor$ or $\lceil x_t(u)/d^+ \rceil$ many tokens in every round.

Observation 3.2. *The algorithm* SEND $([x/d^+])$ *is a good $(d^+ - 2d)$-balancer for $d^+ > 2d$. Furthermore,* ROTOR-ROUTER$*$ *is a good 1-balancer.*

The following theorem shows that good s-balancers achieve a smaller discrepancy of $O(d)$ if they are allowed to run longer than T steps.

Theorem 3.3. *Let G be any d-regular input graph and let d^+ is the degree of the balancing graph G^+. Let δ be an arbitrary constant and let $1 \leq s \leq (d^+ - d)$. Assume A is a good s-balancer. Then A achieves a discrepancy of $((2\delta + 1)d^+ + 4d^\circ)$ after time $\mathcal{O}\left(\log K + ds^{-1} \cdot \log^2 n/\mu\right)$.*

We note that large values of s ($s = \Omega(d)$) increase the speed of the balancing process. In Theorem 4.2 (Section 4) we provide a lower bound of $\Omega(d)$ on the discrepancy of any stateless algorithms, the bound is independent of the balancing time. Since the class of good s-balancers contains many stateless algorithms, this also means that the bound on the discrepancy in Theorem 3.3 cannot be improved without further restrictions on the class.

The remainder of this section is devoted to a proof of Theorem 3.3. We first define the following two families of potential functions, parameterized by c:

$$\phi_t(c) = \sum_{v \in V} \max\{x_t(v) - cd^+, 0\} \text{ and } \phi'_t(c) = \sum_{v \in V} \max\{cd^+ + s - x_t(v), 0\}.$$

To show the theorem we use Equation 7 of the proof of Theorem 2.3, to derive Lemma 3.4. The lemma shows that, for every node u, there exists a time step t_u in which the load of the node has a certain distance to $\bar{x}$. We will then show that the time step t_u results in a potential drop of $\phi_t(c)$ for u if the load of u was larger than cd^+.

The following lemma gives a bound on the required length of the time interval so that there is a step t_u where u has a load which is sufficiently close to $\bar{x}$. The required time is expressed as a fraction of $\log n/\mu$. The lemma shows a tradeoff (parameter λ) between the required time and the load difference of u to $\bar{x}$. The proof can be found in Appendix B.

Lemma 3.4. *Consider any cumulatively δ-fair balancer with remainder bounded by r, and an initialization of the load balancing process with average load $\bar{x}$ and initial discrepancy K. Let $\lambda \geq 0$, and let $t \geq 16 \cdot \log(nK)/\mu$, and let $\widehat{T} = O\left(d \log n/(\mu \cdot (\lambda + 1))\right)$. Then we have:*
For all $u \in V$ there exists a time step $t' \in [t+1; t+\widehat{T}]$ such that $x_{t'}(u) \leq \bar{x} + \delta d^+ + 2r + 1/2 + \lambda$.

The next lemma bounds the one-step potential drop of $\phi_t(c)$ occurring on every node which has a load of more than cd^+ at time $t-1$ and has a smaller load of at most $cd^+ + s$ at time t. The proof can be found in Appendix B.

Lemma 3.5 (Monotonicity of potential)**.** *Let A be a good s-balancer. The potential $\phi_t(c)$ is non-increasing in time and it satisfies: $\phi_t(c) \leq \phi_{t-1}(c) - \sum_{u \in V} \Delta_t(c,u)$, where:*

$$\Delta_t(c,u) = \begin{cases} \min\{x_{t-1}(u), cd^+ + s\} - \max\{x_t(u), cd^+\} & \text{if } x_{t-1}(u) > x_t(u) \text{ and } x_{t-1}(u) > cd^+ \\ & \quad \text{and } x_t(u) < cd^+ + s \\ 0 & otherwise \end{cases}$$

The following observation extends Lemma 3.5 to intervals $[t, t']$. It estimates the potential drop of $\phi_t(c)$ for nodes which have a load $\geq cd^+$ at time t and a load $\leq cd^+$ during one time step of the interval. The observation follows directly from Lemma 3.5; we omit its proof.

Observation 3.6. *Let A be a good s-balancer and let $t \leq t'$ be two fixed time steps. Denote by U the subset of nodes such that for all $u \in U$ $x_t(u) \geq cd^+ + 1$ and there exists a moment of time $t_u \in [t, t']$ such that $x_{t_u}(u) \leq cd^+$. Then $\phi_{t'}(c) \leq \phi_t(c) - \sum_{u \in U} \max\{s, x_t(u) - cd^+\}$.*

The potential defined in Lemma 3.5 bounds the number of tokens above certain thresholds. Now we use $\phi'_t(c)$ to show symmetric results measuring the number of 'gaps' below certain thresholds. The proof of Lemma 3.7 is very similar to that of Lemma 3.5, and we provide it for completeness in Appendix B.

Lemma 3.7. *Let A be a good s-balancer. The potential $\phi'_t(c)$ is non-increasing in time and it satisfies: $\phi'_t(c) \leq \phi'_{t-1}(c) - \sum_{u \in V} \Delta'_t(c,u)$, where:*

$$\Delta'_t(c,u) = \begin{cases} \min\{x_t(u), cd^+ + s\} - \max\{x_{t-1}(u), cd^+\} & \text{if } x_{t-1}(u) < x_t(u) \text{ and } x_{t-1}(u) < cd^+ + s \\ & \quad \text{and } x_t(u) > cd^+ \\ 0 & otherwise \end{cases}$$

Before proving the lemma, we remark that the potential admits a drop at node u at time t (*i.e.*, $\Delta'_t(c,u) \geq 1$) for every node u such that $x_{t-1}(u) \leq cd^+$ and $x_t(u) \geq cd^+ + 1$, for any algorithm which is at least 1-self-preferring. Again, the following observation follows directly from Lemma 3.7; we omit its proof.

Observation 3.8. *Let A be a good s-balancer and let $t \leq t'$ be two fixed time steps. Denote by U the subset of nodes such that for all $u \in U$ $x_t(u) < cd^+ + s$ and there exists a moment of time $t_u \in [t, t']$ such that $x_{t_u}(u) \geq cd^+ + s$. Then $\phi'_{t'}(c) \leq \phi_t(c) - \sum_{u \in U} \max\{s, cd^+ + s - x_t(u)\}$.*

The main idea of the rest of the proof is the following. We will consider the potential functions $\phi_t(c)$ for decreasing values of c and analyze the time T_c it takes to decrease the potentials $\phi_t(c)$. The time bound of Theorem 3.3 is then the sum of the times T_c for suitably chosen values of c. A symmetrical argument can be used to bound $\phi'_t(c)$. Details of the arguments of the proof are provided in Appendix B.

4 Lower Bounds

We start by showing that the cumulative fairness bounds we introduce cannot be completely discarded when improving upon the discrepancy gaps from [17]. Note that a round-fair balancer is not necessarily cumulatively δ-fair for any constant δ. In the following we show that there are round-fair balancers which have a discrepancy of at least $\Omega(\operatorname{diam}(G) \cdot d)$. The proofs of this section are deferred to Appendix C.

Theorem 4.1. *Let G be a d-regular graph. There exists an initial distribution of tokens and a round-fair balancer A, such that A cannot achieve a discrepancy better than $(c \cdot \operatorname{diam}(G) \cdot d)$, for some positive constant $c > 0$.*

The following bound shows that the stateless algorithms we design are asymptotically the best possible in terms of eventual discrepancy. Namely, any stateless algorithm is not able to achieve a discrepancy better than cd, for some constant c. This also means that the bound on the discrepancy, presented in Theorem 3.3, cannot be improved in general for the class of good s-balancers.

Theorem 4.2. *Let A be an arbitrary deterministic and stateless algorithm. For every even n, there exists a d-regular graph and an initial load distribution such that A cannot achieve discrepancy better than cd, for some positive constant $c > 0$.*

Our final lower bounds concern variants of the ROTOR-ROUTER. The next theorem shows that for a graph without self-loops (*i.e.*, $G = G^+$) the best possible discrepancy of the ROTOR-ROUTER is at least $c \cdot d \cdot \varphi'(G)$, where $\varphi'(G)$ is the odd girth of graph G, *i.e.*, the length of the shortest odd length cycle over all nodes of G. This gives for an odd-length cycle of n nodes a discrepancy of at least $c \cdot n$ for some constant c.

Theorem 4.3. *Let G be any d-regular and non-bipartite graph, and let $d^+ = d$. Then, there exists an initial load distribution and direction of the rotors such that* ROTOR-ROUTER *cannot achieve discrepancy better than $(c \cdot d\varphi(G))$, for some positive constant $c > 0$, where $(2\varphi(G) + 1)$ is the odd girth of G.*

5 Conclusion

We introduced two classes of deterministic load-balancing algorithms: Cumulative δ-fair balancers and good s-balancer. The lower bounds show discrepancies of $\Omega(d \cdot \mathrm{diam})$ for algorithms which are not cumulatively δ-fair or which do not have any self-loops. However, there are two main questions which we leave unanswered: 1) How many self-loops are necessary to obtain our bounds? 2) Are the restrictions imposed by good s-balancers necessary to obtain our bounds?

References

[1] C. Adolphs and P. Berenbrink. *Distributed self-prefering load balancing with weights and speeds.* In *PODC*, pages 135–144, 2012.

[2] C. Adolphs and P. Berenbrink. *Improved bounds on diffusion load balancing.* In *IPDPS*, 2012.

[3] H. Akbari and P. Berenbrink. *Parallel rotor walks on finite graphs and applications in discrete load balancing.* In *SPAA*, pages 186–195, 2013.

[4] H. Akbari, P. Berenbrink, and T. Sauerwald. *A simple approach for adapting continuous load balancing processes to discrete settings.* In *PODC*, pages 271–280, 2012.

[5] P. Berenbrink, C. Cooper, T. Friedetzky, T. Friedrich, and T. Sauerwald. *Randomized diffusion for indivisible loads.* In *SODA*, pages 429–439, 2011.

[6] J. Cooper, B. Doerr, T. Friedrich, and J. Spencer. Deterministic random walks on regular trees. *Random Struct. Algorithms*, 37(3):353–366, 2010.

[7] J. Cooper and J. Spencer. Simulating a random walk with constant error. *Combinatorics, Probability and Computing*, 15:815–822, 2006.

[8] B. Doerr and T. Friedrich. Deterministic random walks on the two-dimensional grid. *Combinatorics, Probability and Computing*, 18(1-2):123–144, 2009.

[9] T. Friedrich, M. Gairing, and T. Sauerwald. *Quasirandom load balancing.* In *SODA*, pages 1620–1629, 2010.

[10] T. Friedrich and T. Sauerwald. *Near-perfect load balancing by randomized rounding.* In *STOC*, pages 121–130, 2009.

[11] T. Friedrich and T. Sauerwald. The cover time of deterministic random walks. In *COCOON*, pages 130–139, 2010.

[12] S. Kijima, K. Koga, and K. Makino. *Deterministic random walks on finite graphs.* In *ANALCO*, pages 16–25, 2012.

[13] A. Kosowski and D. Pająk. *Does Adding More Agents Make a Difference? A Case Study of Cover Time for the Rotor-Router.* In *ICALP*, pages 544–555, 2014.

[14] D. Levin, Y. Peres, and E. Wilmer. *Markov chains and mixing times*. *American Mathematical Society, 2006.*

[15] J. Norris, Y. Peres, A. Zhai. *Surprise probabilities in Markov chains*. In *SODA*, pages 1759–1773, 2015.

[16] V. Priezzhev, D. Dhar, A. Dhar, and S. Krishnamurthy. *Eulerian walkers as a model of self-organized criticality. Phys. Rev. Lett.*, 77:5079–5082, 1996.

[17] Y. Rabani, A. Sinclair, and R. Wanka. *Local divergence of Markov chains and the analysis of iterative load-balancing schemes.* In *FOCS*, pages 694–703, 1998.

[18] T. Sauerwald and H. Sun. *Tight bounds for randomized load balancing on arbitrary network topologies.* In *FOCS*, pages 341–350, 2012. A revised and extended version is available online at: http://arxiv.org/abs/1201.2715.

[19] R. Subramanian and I. Scherson. *An analysis of diffusive load-balancing.* In *SPAA*, pages 220–225, 1994.

[20] T. Shiraga, Y. Yamauchi, S. Kijimam, and M. Yamashita. *Deterministic Random Walks for Rapidly Mixing Chains.* `arXiv:1311.3749`, 2013.

[21] V. Yanovski, I. Wagner, and A. Bruckstein. *A Distributed Ant Algorithm for Efficiently Patrolling a Network Algorithmica*, 37(3):165-186, 2003.

APPENDIX

A Omitted Proofs from Section 2

We start with a technical lemma giving bounds on the behaviour of the error matrix $\mathbf{\Lambda}_\tau$ for $\tau \in \mathbf{N}$ with $\tau > \log n/\mu$. We recall that $\mathbf{\Lambda}_t = \mathbf{P}^t - \mathbf{P}^\infty$.

Lemma A.1. *Let $(\mathbf{q}_t)$ be a sequence of vectors parameterized by t, let $\bar{\mathbf{q}}_t = \mathbf{P}^\infty \mathbf{q}_t$, and let $c \in \mathbf{N}$ be an arbitrary constant.*

(i) For $t \geq c \cdot 4\frac{\log(n \cdot \max_\tau \|\mathbf{q}_\tau - \bar{\mathbf{q}}_\tau\|_\infty)}{\mu}$ we have

$$\|\mathbf{\Lambda}_t \mathbf{q}_t\|_\infty \leq 2^{-c}.$$

(ii) The following bound holds:

$$\sum_{t \geq \frac{6c \log n}{\mu}} \|\mathbf{\Lambda}_t \mathbf{q}_t\|_\infty \leq n^{-c} \cdot \max_{\tau \geq \frac{6c \log n}{\mu}} \{\|\mathbf{q}_\tau\|_\infty\}$$

Proof. The proof proceeds by standard arguments (cf. e.g. [14], Chapter 4); we do not attempt to optimize the constants in the claims. Consider the eigendecomposition of matrix $\mathbf{P} = \mathbf{XLX^{-1}}$, where L is the normalized diagonal matrix of eigenvalues $\text{diag}(1, \lambda_2, \ldots, \lambda_n)$, with $1 - \mu \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$, for $2 \leq i \leq n$. We have $\mathbf{\Lambda_t} = \mathbf{XL'_tX^{-1}}$, where $L'_t = \text{diag}(0, \lambda_2^t, \ldots, \lambda_n^t)$. To show claim (i), we take into account that $\bar{q}_t$ is an eigenvalue of $\mathbf{P}$, so $\mathbf{\Lambda}_t \bar{q}_t = 0$, and we can write:

$$\begin{aligned}\|\mathbf{\Lambda}_t \mathbf{q}_t\|_\infty = \|\mathbf{\Lambda}_t(\mathbf{q}_t - \bar{\mathbf{q}}_\mathbf{t})\|_\infty \leq \|X\|_\infty \left\|L'_t\right\|_\infty \left\|X^{-1}\right\|_\infty \|\mathbf{q}_t - \bar{\mathbf{q}}_\mathbf{t}\|_\infty \leq n^2(1-\mu)^t \|\mathbf{q}_t - \bar{\mathbf{q}}_\mathbf{t}\|_\infty \leq \\ < 2^{-\mu t} n^2 \|\mathbf{q}_t - \bar{\mathbf{q}}_\mathbf{t}\|_\infty .\end{aligned}$$

Claim (i) follows directly. To show Claim (ii), observe that we have:

$$\begin{aligned}\sum_{t=\frac{6c \log n}{\mu}}^{+\infty} \|\mathbf{\Lambda}_t \mathbf{q}_t\|_\infty \leq \sum_{i=0}^{+\infty} \sum_{t=\frac{6(c+i)\log n}{\mu}}^{\frac{6(c+i+1)\log n}{\mu}-1} \|\mathbf{\Lambda}_t \mathbf{q}_t\|_\infty \leq \sum_{i=0}^{+\infty} \left(\frac{6c \log n}{\mu} \cdot n^{-6(c+i)+2}\right) \max_{\tau \geq \frac{6c \log n}{\mu}} \{\|\mathbf{q}_\tau\|_\infty\} \leq \\ \leq 2 \cdot \frac{6c \log n}{\mu} \cdot n^{-6c+2} \max_{\tau \geq \frac{6c \log n}{\mu}} \{\|\mathbf{q}_\tau\|_\infty\} < n^{-c} \max_{\tau \geq \frac{6c \log n}{\mu}} \{\|\mathbf{q}_\tau\|_\infty\}.\end{aligned}$$

□

The following proposition shows that the change in the model (the way the tokens are retained) in the preliminaries of Section 2 in comparison to the original model described in Section 1.3 does change the load sent over any original edge in the graph. In particular, every cumulatively fair balancer can be transformed (by shifting tokens from self-loops to the remainder) into an algorithm guaranteeing cumulative fairness on all edges and that this transformed algorithm sends exactly the same load over original edges in every round.

Proposition A.2. *For any cumulatively δ-fair load-balancing algorithm A, there exists an Algorithm A' with remainder $r \leq d^+$ such that for any t and for all $(u, v) \in E_u^+$*

1. $\left|F_t(u,v) - \frac{F_t^{out}(u)}{d^+}\right| \leq \delta$.
2. *the load sent over (u, v) is the same in A and A'.*

Proof. The reformulation of algorithm A as algorithm A' proceeds as follows. For all edges $e \in E(G)$ (*i.e.*, except for self-loops), in every step A' places the same amount of load on e as A. However, in A' load may be retained on nodes in a different way, being placed in the remainder $r_t(u)$ rather than on self-loops at node $u \in V$. To prove, that it is always possible, we proceed by induction.

Specifically, at a fixed moment of time t, let $f_t(e)$ be the amount of load put on an edge e by algorithm A, and $f'_t(e)$ be the amount of load put on an edge by A', and let $F_t(e)$ and $F'_t(e)$ be the respective cumulative loads for algorithms A and A'. Algorithm A' processes all edges (including self-loops) sequentially and verifies if sending this amount of load along e would satisfy the cumulative fairness condition up to time t with respect to all edges original from u already processed.

Let e_1 be the edge or self-loop that violates the cumulative load property for A', that is there exists an incident edge or self-loop e_2 such that $|(F'_{t-1}(e_1) + f_t(e_1)) - (F'_{t-1}(e_2) + f_t(e_2))| > \delta$. Since $|f_t(e_1) - f_t(e_2)| \leq 1$, and $|F'_{t-1}(e_1) - F'_{t-1}(e_2)| \leq \delta$ (from inductive assumption), we get that

$$(F'_{t-1}(e_1) + f_t(e_1)) - (F'_{t-1}(e_2) + f_t(e_2)) \in \{\delta + 1, -\delta - 1\} \tag{10}$$

(without loss of generality we can assume that this value is $\delta + 1$). Moreover, we can show that for every e'_2 such that the pair e_1, e'_2 violates cumulative fairness, the value (10) is $\delta + 1$ (otherwise $F'_{t-1}(e_1) - F'_{t-1}(e_2) = \delta$ and $F'_{t-1}(e_1) - F'_{t-1}(e'_2) = -\delta$ imply $F'_{t-1}(e'_2) - F'_{t-1}(e_2) = 2\delta$ which contradicts the inductive assumption). We can also observe that e_1 is a loop (attached to vertex u), since non-loop edges satisfy cumulatively fairness for A. Thus it is enough to set $f'_t(e_1) = f_t(e_1) - 1$ and increase $r_t(u)$ by one (in the mirror scenario with a value of $-\delta - 1$ in (10) we would set $f'_t(e_1) = f_t(e_1) + 1$ and decrease $r_t(u)$ by one). It is easy to observe that this makes e_1 satisfy δ-fairness with every other edge incident to u. After processing all edges and self-loops and edges in this way and since every edge receives $\lfloor x_t(u)/d^+ \rfloor$ tokens, we eventually obtain that cumulative fairness is preserved, and moreover $|r'_t(u)| \leq d^+$. □

A.1 Proof of Theorem 2.3

Here we finish the proof of Theorem 2.3 presented in Section 2.

Proof. We begin by providing the details of bound (8). Let w be the i'th node of V, then we define $\mathbf{w}$ to be the vector, such that $\mathbf{w}[i] = 1$ and $\forall_{j \neq i} \mathbf{w}[j] = 0$. Let $a = t - \tau$. Then, we have:

$$\begin{aligned}
\left\|(\mathbf{P}^{t+1-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau\right\|_\infty &= \left\|(\mathbf{P}^{a+1} - \mathbf{P}^a)\boldsymbol{\varepsilon}_{t-a}\right\|_\infty = \max_{w \in V} \left|\mathbf{w}^\top \left(\mathbf{P}^{a+1} - \mathbf{P}^a\right) \boldsymbol{\varepsilon}_{t-a}\right| \\
&= \max_{w \in V} \left|\mathbf{w}^\top \left(\mathbf{P}^{a+1} - \mathbf{P}^a\right) \left(\sum_{v \in V} \boldsymbol{\varepsilon}_{t-a}(v)\mathbf{v}\right)\right| \\
&\leq \|\boldsymbol{\varepsilon}_{t-a}\|_\infty \cdot \max_{w \in V} \sum_{v \in V} \left|\mathbf{w}^\top (\mathbf{P}^{a+1} - \mathbf{P}^a)\mathbf{v}\right| \\
&\leq (\delta d^+ + r) \cdot \max_{w \in V} \sum_{v \in V} |P_{a+1}(v, w) - P_a(v, w)| \qquad (8)
\end{aligned}$$

Since the graph is regular, we have $P_t(v, w) = P_t(w, v)$.[1] From here on, we split the analysis of the claims of Theorem 2.3, proving each one of them by bounding (8) separately.

(i) $\mathcal{O}\left((\delta + 1)d\sqrt{\frac{\log n}{\mu}}\right)$**-discrepancy for** $d^+ \geq 2d$ **:**
For regular graphs having $P(u, u) \geq 1/2$ for all $u \in V$, we have by [14] (for $a > 0$)

$$\forall_{w \in V} \sum_{v \in V} |P_{a+1}(w, v) - P_a(w, v)| < \frac{24}{\sqrt{a}}.$$

[1] For general graphs, one can use $P_t(v, w) = (d^+(w)/d^+(v))P_t(w, v)$

(For case of $a = 0$, we have $\forall_{w\in V} \sum_{v\in V} |P_1(w,v) - P_0(w,v)| \leq 2$.)

We obtain by applying $\sum_{a=1}^{t-t^*} 1/\sqrt{a} \leq 2\sqrt{t-t^*}$

$$\sum_{0\leq a<t-t^*} \left\|(\mathbf{P}^{a+1} - \mathbf{P}^a)\boldsymbol{\varepsilon}_{t-a}\right\|_\infty < (\delta d^+ + r)\left(2 + 48\sqrt{\underbrace{t-t^*}_{4t_\mu}}\right) \leq 98(\delta d^+ + r)\sqrt{t_\mu}.$$

Introducing this into (7) and setting $\widehat{T} = 1$ yields

$$\|\mathbf{x}_{t+1} - \bar{x}\|_\infty = \mathcal{O}\left((\delta d^+ + r)\sqrt{t_\mu}\right) = \mathcal{O}\left(((\delta+1)d)\sqrt{\frac{\log n}{\mu}}\right)$$

where we take into account that $t_\mu = 6\log n/\mu$, and that $r \leq d^+$. (Observe that whenever a cumulatively fair balancer has a $\delta = 0$ the bound on the remainder r has to be of order $\Omega(d)$.)

(ii) $\mathcal{O}\left((\delta+1)d\sqrt{n}\right)$**-discrepancy for** $d^+ \geq 2d$ **:**
Let D_{a+1} be the diagonal matrix of $(P_{a+1} - P_a)$ and let X be the corresponding base change matrix.

$$\begin{aligned}
\max_{w\in V} \sum_{v\in V} |P_{a+1}(w,v) - P_a(w,v)| &= \max_{w\in V} \sum_{v\in V} |\mathbf{v}(P_{a+1} - P_a)\mathbf{w}| \\
&= \max_{w\in V} \|(P_{a+1} - P_a)\mathbf{w}\|_1 \\
&\leq \max_{\|\mathbf{w}\|_2=1} \|(P_{a+1} - P_a)\mathbf{w}\|_1 \\
&= \max_{\|\mathbf{w}\|_2=1} \left\|X^\top D_{a+1} X\mathbf{w}\right\|_1 \\
&\leq \sqrt{n}\left\|X^\top D_{a+1} X\right\|_2 \\
&= \sqrt{n}\left\|D_{a+1}X\right\|_2. \qquad (11)
\end{aligned}$$

We have

$$D_{a+1} = \begin{pmatrix} \lambda_1^{a+1} - \lambda_1^a & 0 & \dots & 0 \\ 0 & \lambda_2^{a+1} - \lambda_2^a & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n^{a+1} - \lambda_n^a \end{pmatrix}.$$

where $\lambda_1, \dots, \lambda_n$ are the eigenvalues of P. We note that $\lambda_1 = 1$ and $\lambda_2, \dots, \lambda_n \in [0,1]$ since $d^\circ \geq d$. Hence by plugging (11) in (8) we derive

$$\begin{aligned}
\sum_{0\leq a<t-t^*} \left\|(\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau\right\|_\infty &\leq (\delta+1)d^+ \cdot \sum_{a<t-t^*} \max_{w\in V} \sum_{v\in V} |P_{a+1}(w,v) - P_a(w,v)| \\
&\leq (\delta+1)d^+ \cdot \sum_{a<t-t^*} \sqrt{n}\left\|D_{a+1}X\right\|_2 \\
&\leq (\delta+1)d^+ \cdot \sqrt{n} \sum_{a=0}^{t-t^*} |\lambda_2^{a+1} - \lambda_2^a| \\
&= (\delta+1)d^+ \cdot \sqrt{n} \cdot (\lambda_2^0 - \lambda_2^{t-t^*}) \\
&\leq (\delta+1)d^+ \cdot \sqrt{n}
\end{aligned}$$

Introducing this into (7) and setting $\widehat{T} = 1$ yields:

$$\|\mathbf{x}_{t+1} - \bar{x}\|_\infty \leq (\delta+1)d^+ \cdot \sqrt{n},$$

which completes the proof.

(iii) $\mathcal{O}\left((\delta+1)\frac{d\log n}{\mu}\right)$**-discrepancy for** $d^+ \geq d+1$**:**
We bound the term $\sum_{t^*<\tau\leq t}\left\|(\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau\right\|_\infty$ of (7):

$$\begin{aligned}\sum_{t^*<\tau\leq t}\left\|(\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau\right\|_\infty &\leq \sum_{t^*<\tau\leq t}\left(\left\|(\mathbf{P}^{t+\widehat{T}-\tau}\boldsymbol{\varepsilon}_\tau\right\|_\infty + \left\|\mathbf{P}^{t-\tau}\boldsymbol{\varepsilon}_\tau\right\|_\infty\right)\\ &\leq 2\sum_{t^*<\tau\leq t}\|\boldsymbol{\varepsilon}_\tau\|_\infty\\ &\leq 2(\delta d^+ + r)(t-t^*) = 8t_\mu(\delta d^+ + r).\end{aligned}$$

Putting this into (7) and dividing by $\widehat{T}$ gives:

$$\left\|\frac{\sum_{t<\tau\leq t+\widehat{T}}\mathbf{x}_\tau}{\widehat{T}} - \bar{x}\right\|_\infty \leq \left(\delta d^+ + 2r + \frac{1}{4}\right) + \frac{(8t_\mu+1)(\delta d^+ + r)}{\widehat{T}}. \tag{12}$$

The claim follows from (12) by setting $\widehat{T} = 1$.

□

B Omitted Proofs from Section 3

B.1 Proof of Lemma 3.4

Proof. We build upon the proof of Theorem 2.3 (see Section 2). We will use the notation established in the proof of Theorem 2.3. Since $d^\circ = O(d^+)$, $\delta = O(1)$, and $r \leq d^+$, there is a constant c such that $cd \geq \delta d^+ + r$. We also have $t_\mu = 6\log n/\mu$. We set $\widehat{T} \geq 216c \cdot \frac{d\log n}{\mu(\lambda+1)}$.
We derive from (12)

$$\left\|\frac{\sum_{t<\tau\leq t+\widehat{T}}\mathbf{x}_\tau}{\widehat{T}} - \bar{x}\right\|_\infty \leq \delta d^+ + 2r + \frac{1}{4} + \frac{9t_\mu\cdot cd}{216c\cdot\frac{d\log(n)}{\mu(\lambda+1)}} \leq \delta d^+ + 2r + \frac{1}{4} + \frac{(\lambda+1)}{4} \leq \delta d^+ + 2r + \frac{1}{4} + \lambda + \frac{1}{4}.$$

Therefore, we have that the difference between the average load of any node over $\widehat{T} = \frac{6d\log(n)}{\mu(\lambda+1)}$ steps and the average load $\bar{x}$ is bounded by $\delta d^+ + 2r + \frac{1}{2} + \lambda$. This means that for every node u there has to be a time step $t' \in [t+1, t+\widehat{T}]$ such that $x_{t'}(u) - \bar{x} \leq \delta d^+ + 2r + \frac{1}{2} + \lambda$. This yields the claim. □

B.2 Proof of Lemma 3.5

Proof. Fix $c \in \mathbf{N}$. At any time t, we will divide the set L of m tokens circulating in the system into two groups: the set of *black* tokens L_t^- and the set of *red* tokens L_t^+, with $L = L_t^- \cup L_t^+$. Colors of tokens persist over time unless they are explicitly recolored. For $t = 1$, for each node u we color exactly $|L_1^-(u)| = \min\{x_1(u), cd^+\}$ tokens at u black, and the remaining tokens at u red. In every time step, we follow two rules concerning token distribution:

(1) The number of black tokens leaving a node u along any edge (including self-loops) is never more than c.

(2) At the start of each subsequent step t, we recolor some red tokens to black, so that the total number of black tokens located at a node u is exactly $|L_t^-(u)| = \min\{x_t(u), cd^+\}$.

We note that both rules of token circulation are well defined. The proof proceeds by induction. To prove the correctness of rule (1) at step t, observe that, by the definition of good s-balancers, for any node u we either have $x_t(u) \leq cd^+$ and then node u sends at most c tokens along each of its edges and self-loops, or $x_t(u) > cd^+$, and then node u sends at least c tokens along each of its edges and self-loops. In the first case, rule (1) is correct regardless of how u distributes tokens of different colors; in the second case, u has exactly cd^+ black tokens, and we can require that it sends exactly c of its black tokens along each of its edges and self-loops. To prove the correctness of rule (2), we note that by the correctness of rule (1) for the preceding time step, the number of black tokens arriving at u along edges and self-loops can be upper-bounded by $\min\{x_t(u), cd^+\}$. Hence, no recoloring of tokens from black to red is ever required.

We now observe that the potential $\phi_t(c)$ is by definition the number of red tokens circulating in the system at any given moment of time. Indeed, we have:

$$\phi_t(c) = \sum_{u \in V} (x_t(u) - \min\{x_t(u), cd^+\}) = m - \sum_{u \in V} |L_t^-(u)| = m - |L_t^-| = |L_t^+|.$$

The monotonicity for the potential follows immediately from the fact that no new red tokens appear in the system. To prove the potential drop, we will show that the number of tokens being recolored from red to black at node u in step t is at least $\Delta_t(c, u)$. Indeed, suppose that at time $t-1$ we had for some node u: $x_{t-1}(u) = cd^+ + i$, for some $i \geq 1$. Then, by definition of the self-preference of algorithm A, at least $c+1$ units of load will be sent on at least $i' = \min\{i, s\}$ self-loops of u in step t. By rule (1) of the token circulation process, each of these self-loops will contain at least one red token. Thus, the number of red tokens arriving at u at time t is at least i'. On the other hand, the number of red tokens remaining after the recoloring at u in step t is precisely $\max\{x_t(u) - cd^+, 0\}$. Thus, the number of tokens recolored from red to black at u, or equivalently the potential drop induced at u, is at least

$$\max\{i' - \max\{x_t(u) - cd^+, 0\}, 0\} = \max\{\min\{x_{t-1}(u) - cd^+, s\} - \max\{x_t(u) - cd^+, 0\}, 0\} \stackrel{\text{def}}{=} \Delta_t(c, u)$$

which yields the claimed potential drop. □

B.3 Proof of Lemma 3.7

Proof. The proof is similar to Lemma 3.5. Fix $c \in \mathbf{N}$. At any time t, we will divide the set L of tokens circulating in the system into two groups: the set of *black* tokens L_t^- and the set of *red* tokens L_t^+, with $L = L_t^- \cup L_t^+$. Colors of tokens persist over time unless they are explicitly recolored. For $t = 1$, for each node u we color exactly $|L_1^-(u)| = \min\{x_1(u), cd^+ + s\}$ tokens at u black, and the remaining tokens at u red. In every time step, we follow two rules concerning token distribution:

(1) The number of black tokens leaving u along original edges is at most c.

(2) At the start of each subsequent step t, we recolor some red tokens to black, so that the total number of black tokens located at a node u is exactly $|L_t^-(u)| = \min\{x_t(u), cd^+ + s\}$.

We note that both rules of token circulation are well defined. The proof proceeds by induction. To prove the correctness of rule (1) at step t, observe that, by the definition of good s-balancers, for any node u we either have $x_t(u) \leq cd^+$ and then node u sends at most c black tokens along each of its edges and self-loops, or

$x_t(u) > cd^+$, and then node u sends over $\max\{\min\{x_t(u) - cd^+, s\}, 0\}$ many self-loops $c+1$ black tokens and c along all other edges. In the first case, rule (1) is correct regardless of how u distributes tokens of different colors; in the second case, let $s' = \max\{\min\{x_t(u) - cd^+, s\}, 0\}$. Node u has exactly $cd^+ + s'$ black tokens, and we can require that it sends exactly $c + 1$ of its black tokens along s' many arbitrary self-loops, since the algorithm is s-self-preferring, and exactly c along each of its other edges. To prove the correctness of rule (2), we note that by the correctness of rule (1) for the preceding time step, the number of black tokens arriving at u along edges and self-loops can be upper-bounded by $\min\{x_t(u), cd^+ + s\}$. Hence, no recoloring of tokens from black to red is ever required.

We now observe that the potential $\phi'_t(c)$ is by definition the number of missing black tokens such that every node has $cd^+ + s$ of them. Indeed, we have:

$$\phi'_t(c) = \sum_{u \in V} (cd^+ + s - \min\{x_t(u), cd^+ + s\}) = (cd^+ + s) \cdot n - \sum_{u \in V} |L_t^-(u)| = (cd^+ + s) \cdot n - |L_t^-|.$$

The monotonicity for the potential follows immediately from the fact that no new red tokens appear in the system. To prove the claimed potential drop, we will show that the number of tokens being recolored from red to black in time step t is at least $\Delta'_t(c, u)$. Note, that a red token, which is recolored in black, will decrease the potential by 1.

Indeed, suppose that at time $t-1$ we had for a node u: $x_{t-1}(u) = cd^+ + s - i$, for an integer $i \geq 1$. Then, by the definition of the self-preference of algorithm A, at least $i' = \min\{i, s\}$ self-loops carry at most c tokens in step t. Intuitively, each of them can 'trap' a red token. By rule (1) of the token circulation process, every neighbor of u sent at most c black tokens. Thus, the number of black tokens arriving at u at time t is at most $cd^+ + s - i'$, and the number of red tokens which u receives is at least $\max\{x_t(u) - (cd^+ + s - i'), 0\}$. Therefore, for $x_{t-1}(u) < cd^+ + s$, since at most i' self-loops 'trap' a red token we have that the number of red tokens which are repainted black at node u at time t is at least (by rule (2) of recoloring)

$$\begin{aligned}
&\min\{\max\{x_t(u) - (cd^+ + s - i'), 0\}, i'\} \\
&= \min\{\max\{\ x_t(u) - (cd^+ - \min\{cd^+ - x_{t-1}(u), 0\}), 0\}, \min\{cd^+ + s - x_{t-1}(u), s\}\} \\
&= \max\{\min\{x_t(u) - x_{t-1}(u), s, x_t(u) - cd^+, cd^+ + s - x_{t-1}(u)\}, 0\} = \Delta'_t(c, u),
\end{aligned}$$

and for $x_{t-1}(u) \geq cd^+ + s$ we have $\Delta'_t(c, u) = 0$, which yields the claimed potential drop.

□

B.4 Proof of Theorem 3.3

Proof. We consider the behavior of the process for $t \geq T = \mathcal{O}(\frac{\log(Kn)}{\mu})$. Due to the monotonicity described in Lemma 3.5, once the maximum load in the system is below cd^+ for some c, it will stay below this threshold. In particular, this means that the maximum load will not exceed $x_m = \bar{x} + \mathcal{O}\left(\frac{d \log n}{\mu}\right)$ after $\mathcal{O}\left((\delta + 1)d \cdot \log n/\mu\right)$ time, as shown in Theorem 2.3(iii).

In the first part of the proof, we will show that after a further $\mathcal{O}(\frac{d^+}{s}\frac{\log^2 n}{\mu})$ time steps, the maximum load in the network will drop below the threshold value $c_0 d^+$, where c_0 is the smallest integer such that $c_0 d^+ \geq \bar{x} + \delta d^+ + 2d^\circ + d^+/2$.

We note that if there exists a node u with load $x_t(u) \geq c_0 d^+ + 1$ at some time moment $t \geq T$, then by Lemma 3.4 with $\lambda = d^+/2 - 1/2$ (which we can apply to good s-balancers, putting $r = d^\circ$, by Proposition A.2), there exists some time step $t' \in [t+1; t+\widehat{T}]$, where $\widehat{T} = \mathcal{O}\left(\frac{\log n}{\mu}\right)$, such that we have $x_{t'}(u) \leq c_0 d^+$. We then obtain directly from Observation 3.6 that such a load change for node u results in the decrease of potential $\phi(c_0)$ in the time interval $[t+1; t+\widehat{T}]$. Since the potential $\phi(c_0)$ is non-increasing and non-negative, it follows that eventually the load of all nodes must be below the threshold $c_0 d^+$.

In order to prove that the load of all nodes drop below $c_0 d^+$ within $\mathcal{O}(\frac{d^+}{s}\frac{\log^2 n}{\mu})$ time steps after time T, we apply a more involved potential-decrease argument based on the parallel drop of multiple potentials $\phi(c)$, for $c \in \{c_0, c_0+1, \ldots, c_1\}$. Here, c_1 is the smallest integer such that $c_1 d^+ \geq x_m = \bar{x} + \mathcal{O}\left(\frac{d \log n}{\mu}\right)$.

We now partition the execution of our process into phases of duration t_p, $p = 1, 2, 3, \ldots, p_f$, such that, at the end of moment $T_p = T + t_1 + \ldots + t_p$ (end of the p'th phase), the following condition is satisfied:

$$\phi_{T_p}(c) \leq 4^{(c_1-c)} 2^{-p} (c_1 - c_0) d^+ n, \quad \text{for all } c_0 \leq c \leq c_1. \tag{13}$$

As we have remarked, the values of time t_p are well defined, since eventually the potentials $\phi(c)$ drop to 0, for all $c_0 \leq c \leq c_1$. Our goal is now to bound the ending time T_{p_f} of phase p_f, where:

$$p_f = 2(c_1 - c_0) + \lceil \log((c_1 - c_0) d^+ n) \rceil + 1. \tag{14}$$

At the end of this phase, we will have by (13) that $\phi_{T_{p_f}}(c_0) \leq 1/2$, hence $\phi_{T_{p_f}}(c_0) = 0$, and so there are no nodes having load exceeding $c_0 d^+$.

We now proceed to show the following bounds on the duration of each phase t_p:

$$t_p = \mathcal{O}\left(\frac{d^+}{s} \frac{d \log n}{\mu \cdot \max\{(2(c_1 - c_0) - p) d^+ + 1, d^+/2 + 1\}}\right). \tag{15}$$

For any fixed p, assume that bound (15) holds for all phases before p. For phase p, the proof proceeds by induction with respect to c, in decreasing order of values: $c = c_1, c_1 - 1, \ldots, c_0$.

First, we consider values of $c \geq c_1 - p/2$. For a fixed c, following (13) we denote $b = 4^{(c_1-c)} 2^{-p} (c_1 - c_0) d^+ n$. Knowing that $\phi_{T_{p-1}}(c) \leq 2b$, $\phi_{T_{p-1}}(c+1) \leq b/2$, and by the inductive assumption $\phi_{T_p}(c+1) \leq b/4$, we will show that $\phi_{T_p}(c) \leq b$. Let $s_t(c) := \phi_t(c) - \phi_t(c+1)$; intuitively, $s_t(c)$ can be seen as total the number of tokens in the system which are "stacked" on their respective nodes at heights between $cd^+ + 1$ and $(c+1)d^+$. For $t \in [T_{p-1}; T_p]$, $s_t(c)$ satisfies the following bound:

$$s_t(c) = \phi_t(c) - \phi_t(c+1) \geq \phi_t(c) - \phi_{T_{p-1}}(c+1) \geq \phi_t(c) - \tfrac{b}{2}. \tag{16}$$

For any such time moment t consider the set of nodes with load at least cd^+ at time t. Within the time interval $[t+1; t+\widehat{T}]$, where the period of time $\widehat{T} = \mathcal{O}\left(\frac{d \log n}{\mu \max\{(2(c-c_0)d^+ + 1), d^+/2 + 1\}}\right)$ follows from Lemma 3.4, every node with a load of more than cd^+ at time t, will decrease its load below $\bar{x} + \delta d^+ + 2d^\circ + \frac{1}{2} + \frac{1}{2}\max\{(2(c - c_0)d^+), d^+/2\} \leq c_0 d^+ - d^+/2 + \frac{1}{2} + \max\{(d^+(c - c_0)), d^+/4\} \leq cd^+ + \frac{1}{2}$ (and so also below cd^+) at some moment of time during the considered time interval. By Observation 3.6 a potential drop occurs for $\phi(c)$ in the considered interval $[t+1; t+\widehat{T}]$. More precisely, every node u with $x_t(u) \in [cd^+, cd^+ + s]$ contributes $x_t(u) - cd^+$ to both the potential drop and the value of $s_t(c)$, whereas every node u with $x_t(u) > cd^+ + s$ contributes exactly s to the potential drop and at most d^+ to the value of $s_t(c)$. Hence, we obtain from Observation 3.6 (and the fact that $s \leq d^+$):

$$\phi_{t+\widehat{T}}(c) \leq \phi_t(c) - \frac{s}{d^+} \cdot s_t(c). \tag{17}$$

Combining (16) and (17), we obtain for any time moment $t \in [T_{p-1}, T_p]$:

$$\phi_{t+\widehat{T}}(c) \leq \phi_t(c) - \frac{s}{d^+} \cdot (\phi_t(c) - \tfrac{b}{2}). \tag{18}$$

We can transform this expression to the following form:

$$\phi_{t+\widehat{T}}(c) - \tfrac{b}{2} \leq \phi_t(c) - \tfrac{b}{2} - \frac{s}{d^+} \cdot (\phi_t(c) - \tfrac{b}{2}) = \left(1 - \frac{s}{d^+}\right) \cdot (\phi_t(c) - \tfrac{b}{2}).$$

Observe that we can fix t_p satisfying (15) so that $t_p \geq \frac{d^+}{s}\widehat{T}$ (which implies $T_p \geq T_{p-1} + \frac{d^+}{s}\widehat{T}$) where we took into account that $2(c - c_0) \geq 2(c_1 - c_0) - p$ for $c \geq c_1 - p/2$. Now, taking advantage of the monotonicity of potentials, we have from (18):

$$\phi_{T_p} \leq \phi_{(T_{p-1}+\frac{d^+}{s}\widehat{T})}(c) \leq \tfrac{b}{2} + \left(1 - \frac{s}{d^+}\right)^{\frac{d^+}{s}} \cdot (\phi_{T_{p-1}}(c) - \tfrac{b}{2}) \leq \tfrac{b}{2} + \tfrac{1}{2}(2b - \tfrac{b}{2}) = \tfrac{3}{4}b \leq b,$$

which completes the inductive proof of the bound on t_p for $c \geq c_1 - p/2$.

Moreover, for $c < c_1 - p/2$, (13) holds because $4^{(c_1-c)}2^{-p} > 1$, and $\phi_{T_p}(c) \leq (c_1 - c_0)d^+n$ holds by the definition of potentials.

Now, taking into account (14) and (15), we can bound the time of termination of phase p_f of the process as follows:

$$\begin{aligned}
T_{p_f} &= T + \sum_{p=1}^{p_f} t_{p_f} \\
&= T + \sum_{p=1}^{p_f} \mathcal{O}\left(\frac{d^+}{s}\frac{d\log n}{\mu \cdot \max\{(2(c_1 - c_0) - p)d^+ + 1, d^+/2 + 1\}}\right) \\
&= \mathcal{O}\left(T + \frac{d}{s}\frac{\log n}{\mu}\left(\left(\sum_{p=1}^{2(c_1-c_0)-1} \frac{1}{2(c_1 - c_0) - p + 1/d^+}\right) + \frac{d^+}{d^+/2 + 1} \cdot \log\left((c_1 - c_0)d^+n\right)\right)\right) \\
&= \mathcal{O}\left(T + \frac{d}{s}\frac{\log^2 n}{\mu}\right),
\end{aligned}$$

where we recall that p_f was given by expression (14), and that $c_1d^+ - c_0d^+ = \mathcal{O}\left(\frac{d\log n}{\mu}\right)$.

In this way, we have shown that a balancedness of $(\delta+1/2)d^+ + 2d^\circ$ is achieved in time $\mathcal{O}\left(T + \frac{d}{s}\frac{\log^2 n}{\mu}\right)$. By using the same techniques and Observation 3.8 instead of Observation 3.6, we can show that no node has a load of less than $\bar{x} - (\delta+1/2)d^+ + 2d^\circ$. This gives the desired discrepancy bound of $(2\delta+1)d^+ + 4d^\circ$. □

C Omitted Proofs from Section 4

C.1 Proof of Theorem 4.1

Proof. We will describe an initial state, corresponding to a steady state distribution, such that the flow of load along each edge e is the same in every moment of time ($f_0(e) = f_1(e) = f_2(e) = \ldots$), and the load of nodes does not change in time (however, the load of two distant nodes in the graph will be sufficiently far apart). We take two vertices u and w such that the distance between them is $\mathrm{diam}(G)$. We assign to every node $v \in V$ a value $b(v)$ being the shortest path distance from v to u ($b(u) = 0$, $b(v) = 1$ for direct neighbors of u, etc.). For any given edge (v_1, v_2), we assign

$$f_0(v_1, v_2) = \min(b(v_1), b(v_2))$$

We observe, that for each v:

$$\max_{e_1,e_2 \in E_v} |f_0(e_1) - f_0(e_2)| \leq 1$$

and for each edge (v_1, v_2):

$$f_0(v_1, v_2) = f_0(v_2, v_1).$$

Thus, at each step there exists a way to assign values of $\lceil f(v) \rceil$ and $\lfloor f(v) \rfloor$ so as to achieve desired values over edges, and that the system is in the steady state. The sought value of discrepancy is achieved for the considered pair of nodes u and w whose distance in G is $\mathrm{diam}(G)$. □

C.2 Proof of Theorem 4.2

Proof. We take an arbitrary graph G with n vertices which contains a $\lfloor d/2 \rfloor$-clique C. We can construct such a graph by taking nodes numbered from 0 to $n-1$ and connecting each pair of nodes i and j with an edge if and only if $(i-j) \bmod n \in \{n - \lfloor d/2 \rfloor, \ldots, n-1, 0, 1, \ldots, \lfloor d/2 \rfloor\}$. If d is odd, we also add edges (i,j) for all $(i-j) \bmod n = n/2$. W.l.o.g. we can assume that $C = \{0, 1, \ldots, \lfloor d/2 \rfloor - 1\}$.

Let A be a deterministic and stateless algorithm. Since A is deterministic and stateless, for each node u, at round t the load which A sends to neighbors and the load it keeps through the remainder vector depend solely on the current load of u. Let us fix $\ell = |C| - 1$. Initially, the load is distributed in such a way that every node in C has load ℓ and every other node has load 0. For any given node whose current load is ℓ, let p° denote the number of tokens kept by A at the considered node, and let $p_1, p_2, \ldots, p_d$ be the number of tokens sent by A along respective original edges. Clearly, $\ell = p^\circ + \sum_{i=1}^{d} p_i$. At most ℓ of those values are positive, so we can assume w.l.o.g. that $p_d = p_{d-1} = \ldots = p_{\ell+1} = 0$.

We complete the construction in such a way that at each time step the load over every node is preserved. Let us fix $i \in C$. We design an adversary which has control over which values from $\{p_1, \ldots, p_d\}$ are sent along edges of the clique, and which chooses to send along edges of the clique the possibly nonzero values $p_1, p_2, \ldots, p_\ell$. These values will be assigned to the edges $(i, (i+1) \bmod d), (i, (i+2) \bmod d), \ldots, (i, (i-1) \bmod d)$, respectively. We assign all other values arbitrarily since they are all equal to 0. Thus, we observe that at each step loads of nodes are preserved, since the new load of all nodes having load ℓ at the end of a step is $p^\circ + \sum_{i=1}^{\ell} p_i = \ell$ in the next step. All other nodes in $V \setminus C$ will not receive any tokens and they will remain with load 0. Thus, the load of nodes in the graph does not change over rounds and the load difference of nodes in C and the nodes in $V \setminus C$ is cd, for some constant $c > 0$. □

C.3 Proof of Theorem 4.3

Proof. Let u be an arbitrary vertex belonging to the shortest odd cycle. We assign to every node $v \in V$ a value $b(v)$ being the shortest path distance from v to u ($b(u) = 0$, $b(v) = 1$ for direct neighbors of u, etc.). Observe that for any edge (v_1, v_2), we have $b(v_1) - b(v_2) \in \{-1, 0, 1\}$. Moreover, we have $b(v_1) = b(v_2)$ only if $b(v_1) \geq \varphi(G)$. Suppose $b(v_1) < \varphi(G)$. We can construct an odd length cycle $u \rightsquigarrow v_1 \to v_2 \rightsquigarrow u$ of at most $2 \cdot \varphi(G) - 1$, a contradiction.

For the construction, fix a sufficiently large integer $L > 0$ which will be linked to the average load in the system (it has no effect over the final discrepancy, we need L to be large enough to have nonnegative values of load). We will design a configuration of load in the system which will alternate between two different states, identical for all configurations in odd time steps and even time steps, respectively (thus, $f_0(e) = f_2(e) = \ldots$ and $f_1(e) = f_3(e) = \ldots$). We will describe a configuration at any time step by providing values distributed over every edge. The load distributed over edge (v_1, v_2) will only depend on the values of $b(v_1)$ and $b(v_2)$. If $b(v_1) \geq \varphi(G)$ or $b(v_2) \geq \varphi(G)$, we set $f_0(v_1, v_2) = L$, otherwise:

$$f_0(v_1, v_2) = \begin{cases} L + (\varphi(G) - \min(b(v_1), b(v_2))) & \text{if } 2 | b(v_1) \text{ and } 2 \nmid b(v_2), \\ L - (\varphi(G) - \min(b(v_1), b(v_2))) & \text{if } 2 \nmid b(v_1) \text{ and } 2 | b(v_2). \end{cases}$$

We also set:

$$f_1(v_1, v_2) = f_0(v_2, v_1). \tag{19}$$

Setting all of $f_0(e)$ and $f_1(e)$ is enough to describe every value over every edge. We now prove that such a configuration is possible for some execution of the ROTOR-ROUTER algorithm, *i.e.*, that there exists an ordering of the original edges of each node in the cycle of the ROTOR-ROUTER which leads to such alternating configurations. We observe that $f_t(v_1, v_2) + f_t(v_2, v_1) = 2L$ for $t \in \mathbf{N}$. Thus, for $t \in \mathbf{N}$ we have

$$f_t(v_1, v_2) + f_{t+1}(v_1, v_2) = 2L. \tag{20}$$

We observe, that for any node v and its two neighbors v_1, v_2, it holds

$$|f_t(v, v_1) - f_t(v, v_2)| \leq 1$$

Also, due to (20):

$$\ldots = f_t(v, v_1) - f_t(v, v_2) = -\left(f_{t+1}(v, v_1) - f_{t+1}(v, v_2)\right) = f_{t+2}(v, v_1) - f_{t+2}(v, v_2) = \ldots$$

By (19), the incoming and original flows of load through edges are preserved, and because the difference between original flows is alternating in signs (for two incident original edges), it is always possible to choose an edge ordering in the cycle of the ROTOR-ROUTER representing such a situation. Indeed, we observe that for a particular vertex v, original directed edges of v can be partitioned into two sets $P_1 \cup P_2$, where in even steps edges from P_1 are given one more token than edges from P_2, and in odd steps edges from P_1 are given one less token than edges from P_2. So it is enough to select an ordering of edges for the ROTOR-ROUTER such that every edge from P_1 precedes every edge from P_2 set.

We observe that the node u alternates between loads $(L+\varphi(G))\cdot d$ and $(L-\varphi(G))\cdot d$, while the average load of a node in this setting is exactly $L \cdot d$, which gives us the claimed discrepancy. □